(In print)

The University of Arizona
Engineering College
Department of Electrical Engineering
Computer Science Research Laboratory

CSRL Memo No. 211

A DIGITAL CONTROLLER PROGRAM FOR ANALOG-SIMULATED SYSTEMS (LOCUST/PDP-9 HYBRID)

September, 1970

W. R. Moore

ACKNOWLEDGEMENT:

The writer is grateful to the National Aeronautics and Space Administration for their support of this work under NASA Grant ~~NGR 646~~. NGL-03-002-024

A DIGITAL CONTROLLER PROGRAM FOR ANALOG-SIMULATED SYSTEMS (LOCUST/PDP-9 HYBRID)

INTRODUCTION

This paper describes a PDP-9 assembly language program which allows direct implementation of the z-transform of the transfer function of a linear digital controller of any order up to sixth. This controller is interfaced to the LOCUST hybrid computer through a single ADC/MDAC pair and several control lines. No provision is made for state variable control. The upper bound on the sampling rate is 4khz. due to the conversion time of the ADC used. This maximum is seldom realizable except for systems of less than third order due to the time consumed by the digital program in doing multiplications.

THE CONTROLLER

It is assumed that the transfer function of the digital controller can be written in z-transform notation as

$$\frac{U}{Y} = D(z) \doteq K_d \frac{A_0+A_1 z^{-1}+A_2 z^{-2}+\cdots+A_n z^{-n}}{1+B_1 z^{-1}+B_2 z^{-2}+\cdots+B_m z^{-m}} \quad m \leq n \leq 6 \qquad (1)$$

The block diagram assumed for the closed loop system is shown in Fig. I. (Note the zero-order hold in the MDAC.)

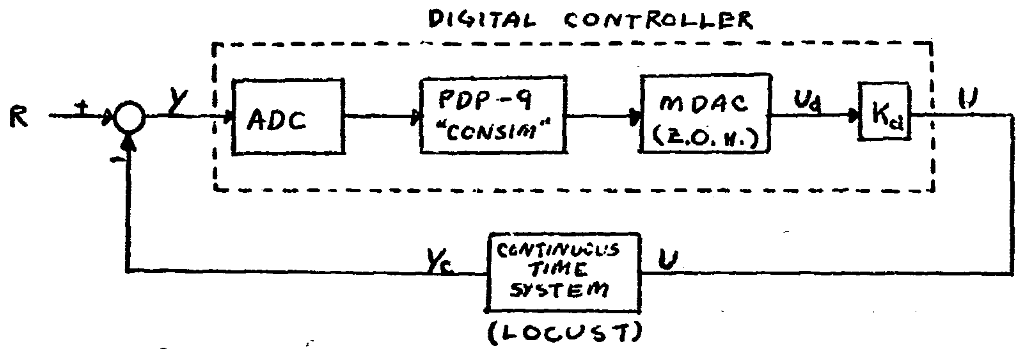


FIG. I. Block Diagram of the Closed Loop System.

The coefficients of the transfer function are entered as single-precision floating point numbers less than or equal to Ø.999999 through the FORTRAN subroutine DIGIN. CONSIM then takes these and un-normalizes the FORTRAN input to 18 bit, ones complement integers. One's complement notation is required by the multiplication hardware in the PDP-9. Any magnitude scale factors for the controller must be introduced in the analog computer through the constant K_d.

CONSIM realizes the transfer function by forming a control, U_d, from the weighted sum obtained from (1) as

$$U_d(k) = (A_\emptyset)Y(k)+(A_1)Y(k-1)+\cdots+(A_n)Y(k-n)$$
$$-(B_1)U_d(k-1)-(B_2)U_d(k-2)-\cdots-(B_m)U_d(k-m). \qquad (2)$$

The values of the Y's and the U_d's are simply saved and multiplied by their respective coefficients. Then Y(k) is made to become Y(k-1), $U_d(k)$ is made to become $U_d(k-1)$, etc. by exchanging the contents of the appropriate registers.

The sequence of operations within the iterative simulation loop of CONSIM is to read ADC #1, multiply by $A\phi$, add this product to previously accumulated products corresponding to the other terms of (2), transfer $U_d(k)$ to MDAC #1 and simultaneously apply the MDAC output to the continuous time system simulated in LOCUST. (Note that ADC #1 and MDAC #1 must be used for these functions. A simple program modification, however, could allow the use of different ADC/MDAC pairs.) The time involved so far is the so-called processing delay which is inherent in the physical realization of (1) or (2). The $U_d(k)$ formed by CONSIM is actually $U_d(k+eps)$, where eps is the processing delay in terms of the sample period. This delay will cause the real controller results to differ from the theoretical predictions by an amount which will increase with the sampling rate. After the transfer of U_d to the MDAC, the saved values of $Y(i)$ and $U_d(i)$ are shifted in time as mentioned above, the other products of (2) are formed and accumulated, ready to be added to the next value of $(A\phi)Y(k)$. Then the conversion complete flag of ADC #1 is repetitively tested until it indicates the ADC is ready with the new $Y(k)$, and the sequence of operations repeats.

CONSIM checks for any coefficients in (2) that have been specified as zero and replaces the first instruction in the multiplication macro for those coefficients by a jump to the next multiplication macro (the last macro jumps back to test the ADC flag) so that time is not wasted multiplying by zero.

This does not apply to the coefficient $A_\emptyset$. Y(k) is always multiplied by $A_\emptyset$ even if it is zero. This saves time in systems where $A_\emptyset \neq \emptyset$ (the usual case), though it takes more time where $A_\emptyset = \emptyset$.

The ADC's in LOCUST are 11 bit, two's complement units; the MDAC's are 12 bit, two's complement, but the least significant bit is not used by CONSIM. The one's complement multiplication of the PDP-9 requires conversion on input/output.

THE ANALOG-SIMULATED SYSTEM

It is assumed that the user of the programs described here is thoroughly familiar with the hybrid computer LOCUST and its PDP-9 interface. Therefore, detailed patching instructions will not be given. (See References 1 and 2.)

The continuous time system may be simulated in real-time on LOCUST by using the upper, "slow" integrator inputs, a 1.∅ microfarad capacitor, and control signal $\overline{\text{IR}}$. Of course time scaling may also be used if the appropriate changes are made in the controller transfer function and sampling rate limitations of the system are kept in mind.

Inputs to the system are best applied by using the relays in LOCUST that are controlled in the LOCUST digital patchbay either by manual push buttons or by levels applied to the control register by CONSIM.

The multiplying feature of the MDAC's is not used and their analog inputs should be patched to +10 and -10 volts.

SAMPLING RATE CONSIDERATIONS

The maximum conversion time required by the ADC's is 25Ø microseconds (for an input of +10 volts). An estimate of the average time required by the actual simulation loop of CONSIM for a third order system is 3ØØ microseconds. Both of these times are dependent upon the magnitude of the problem variables and upon their sign. With a third order system utilizing the full ±10 volt range of LOCUST, a "safe" sampling rate would be about 2 khz. Since CONSIM adjusts itself to eliminate multiplications by zero coefficients, the sampling rate of low order problems is limited by the ADC. In problems with controllers of third or greater order, the digital program consumes more than 25Ø microseconds in doing multiplications and thus limits the sampling rate.

Any of the run-time options described below that are used will cause the simulation loop of CONSIM to take slightly more time and this may have to be considered when using high sampling rates.

PROGRAM OPERATION

Both CONSIM and its FORTRAN subroutine DIGIN must be loaded with the Linking Loader. It is suggested that the core load be saved in disk area QØ following loading. Since frequent spurious interrupts may occur in the initial phases of problem set-up, saving core in QØ allows quick recovery.

The first phase of CONSIM allows the user to choose any of several options which affect the run-time performance of the program. This is done by answering several questions with yes or no (unrecognizable answers cause repetition of the question). The first option is whether to allow digital overflow. If digital overflow is allowed, the most significant bits of any U_d which exceeds 1777_8 in magnitude will be truncated. (1777_8 in the ADC or MDAC corresponds to +10 volts in LOCUST.) Of course this usually does not occur in a conventionally designed controller. If the user desires, the alternative to digital overflow is to force saturation at $\pm 1777_8$ by answering NO to the question "Allow digital overflow?" This does not affect the saved values of $U_d(i)$, only the quantity transmitted to the MDAC. Thus with this option, an input to the continuous time system which causes the controller to temporarily overflow may be recovered from in time. It does, however, add form 1Ø to 16 microseconds to the processing delay.

The second run-time option is to save successive values of the control U_d and the output of the continuous time system Y_c for later display and hardcopy output from the teletype. This option adds no time to the processing delay, but does add five microseconds to the total simulation loop. U_d is saved by use of an auto-index register, requiring only one extra instruction. Since Y_c is not directly available to CONSIM, it was decided that it should be saved by use of the LOCUST data channel. This means that the only programming considerations are that CONSIM must set-up the proper quantities in locations 32 and 33 in the PDP-9 to serve as the word count and current address registers respectively. This is done prior to the beginning of the simulation and consumes no time within the simulation loop. Without further intervention from CONSIM, the data channel "steals cycles" from the PDP-9 to make transfers which take three microseconds each. These three microseconds are added to the time in the simulation loop, but may or may not occur during the processing delay. The data channel must be properly patched so that Y_c is input to an ADC other than #1, that the ADC flag be used to provide the data channel request pulse, and that the ADC data be patched to the 11 low order bits of the data channel data lines.

Note that if it is only desired to save the values of U_d simply do not provide the data channel with a request pusle. (at the time of writing this paper, the LOCUST data channel is not yet complete and none of the results presented here include saved values of Y_c. The necessary set-up of locations 32 and 33 is included in CONSIM, and when the data channel is completed, it should only be necessary to patch it as described above.)

If the save option is elected, CONSIM will ask for a number of points, N_s, to be saved. This may be any non-zero number less than 251.

The final run-time option is to save-on-input. This is possible, of course, only if the user has also chosen the save option just discussed. If save-on-input is not chosen, saving of U_d and Y_c will begin with the first sample time and will continue for N_s-1 sample periods, and the system input, if one is desired, must be applied manually before these N_s-1 sample periods have elapsed. If save-on-input is chosen, CONSIM will place the system into operation with no inputs and repetitively test Free Flag 1 (the second free flag). When FF 1 goes to a "1" CONSIM will begin saving the output variables immediately (or at least as soon as CONSIM detects that FF 1 is a "1"), and after N_I user specified samples have been saved, CONSIM will apply a "1" to control register bits 1-3 which may be used to apply the input to the system. This allows the user to observe the system in operation without input for an unspecified period of time, then start the saving process when he is ready and have N_I samples saved before the input is automatically applied. N_I may be any number less than 251, but obviously should be less than N_s.

It is suggested that for save-on-input, a push button be used to set a flip flop which in turn sets FF 1. Then bit 1 of the control register can be used to control a relay to apply the system input and reset the flip flop connected to FF 1. The other bits are provided for other uses such as gating with the scope sync pulse to provide a good scope picture of the system response.

By electing to not save-on-input, the user may easily observe the response of the system to initial conditions applied in LOCUST.

After run-time options are chosen, the coefficients of the digital controller are entered. DIGIN expects an entry on the CRT keyboard of the form

B2=.327000

where the trailing zeros are not necessary. Actually the format is very free, but care should be taken to not enter a number with more than six digits since the FORTRAN input routine may round such numbers in strange ways. DIGIN will not accept numbers greater than Ø.999999 and will ignore unrecognizable entries. BØ is fixed at Ø.999999 by DIGIN and if the user attempts to change it, the program will inform the user of this fact.

The user actually initiates a simulation by typing the word RUN instead of a coefficient. This returns control to CONSIM which, after some housekeeping and un-normalizing the coefficients to integers, starts the simulation by applying a "1" to control register bit Ø to place LOCUST in Compute mode. This bit should have been previously patched to an OR/NOR gate, the complemented output of which goes to Ext. IR and the uncomplemented output of which goes to Ext. CP on the digital patchbay of LOCUST.

A simulation is terminated by applying a "1" to Free Flag Ø (the first free flag) by means of a manual push button (a flip flop should not be used for this purpose). This causes CONSIM to apply "Ø" to all control register bits, which puts LOCUST in Initial Reset mode. CONSIM then queries the user about any changes he might wish to make in run-time parameters. Then the user may change coefficients and immediately re-run the simulation. If it is desirable to first view the display of the saved variables or to get hard copy output, the user should answer no to the question "Change Coefficients?" the first time it is asked. Then control is returned to DIGIN for saved variable output as chosen by the user. After the output phase is finished, CONSIM will again allow the user to change coefficients and initiate another simulation. If the "Change Ceofficients?" question is again answered with no, CONSIM waits for the response RUN to re-run under the same coefficients or the response NO to terminate CONSIM and return to the PDP-9 Monitor system.

Variables are chosen for display by typing at the appropriate time, the word CONTROL (or just C) for a display of U_d against time or the word OUTPUT (or O) for a display of Y_c against time. Entering the word NONE (or N) will terminate display viewing and begin hard copy output if the user desires. After selecting a display, the user may specify any number of points for display. The number need not correspond with the number of saved points. A full scale value in volts will be displayed on the CRT for each display chosen.

INTERFACE CONTROL FUNCTIONS

The following summary is presented to aid the user in patching the various control signals needed for CONSIM.

CONTROLS FROM CONSIM:

1. Control register bit ∅ = { "1" puts LOCUST in Compute; "∅" puts LOCUST in Reset.

2. If save-on-input chosen, control register bits 1-3 are set to "1" N_T-1 sample periods after the saving is initiated by FF 1. Used to apply system input and other associated functions.

CONTROLS FROM LOCUST DIGITAL PATCHBAY:

1. Signals to CONSIM:

   a. ADC #1 flag (does not have to be patched, but Skip switch must be ON).

   b. FF ∅ signals CONSIM to terminate a simulation.

   c. FF 1 signals CONSIM to initiate save-on-input. (Should be actuated by a flip flop which may be reset by control register bits 1-3.)

2. Signals to Analog Patchbay:

   a. Integrator control should be IR.

   b. ADC control normally taken from appropriate digital clock pulse.

   c. Control register bit ∅ used for Ext. CP and complemented for Ext. IR.

   d. A digital command from a button or from control register bits 1-3 patched to K1, K2, K3, or K4 (relays) and used to apply system input.

   e. R pulse may be patched to Overload Reset terminal but is not necessary.

DATA CHANNEL CONTROL:

1. Appropriate ADC flag (any but #1) patched to RQ terminal.

2. ADC data patched to low order data channel data bits.

3. (Other patching may be necessary when data channel is complete ans working.)

SYSTEM QUIRKS

The hybrid combination of LOCUST and the PDP-9 may play a few tricks on the unfamiliar user. He has already been cautioned to save the core load produced by the Linking Loader in disk area Q∅ in order to quickly recover from spurious interrupts that are likely to occur. The most trouble free procedure to follow in setting up a problem for CONSIM is to first patch the continuous time system on LOCUST and check it out as much as possible before using PDP-9 computer time. This should be done with the PI, Skip, and ADC flag switches all in the OFF position so that whoever is using the PDP-9 will not be bothered by any spurious interrupts (it is even safer to completely unplug the I/O cable from LOCUST when power is down). Once the analog system is checked out and the user's PDP-9 time has begun, make sure that the I/O cables are properly connected to LOCUST, put the Skip switch only in the ON position, and place LOCUST in Initial Reset mode. Then depress IO Reset on the PDP-9 console and restart the system Monitor. This insures that all interface flags in LOCUST are reset before starting the problem.

If at any time the user places LOCUST in the Patch or Potset modes while CONSIM is running, an IOPS Ø3 error (illegal interrupt) will occur and it will be required that CONSIM be reloaded (from QØ) and restarted.

EXAMPLES:

Example 1

Presented as a first example is a simple test for CONSIM which is simple to implement and tests most of the program's functions. The continuous time system is just a sine-loop oscillating at 1 hertz with no damping and an initial condition of 4 volts. The output of this system is patched to ADC #1. The controller is simply a constant multiplier with a phase shift of six sample periods. In z-transform notation

$$D(z) = Ø.5\ z^{-6}.$$

U_d (the output of MDAC #1) is not used as a control, but simply is connected to the scope and strip-chart recorder for observation. Thus U_d should be half the amplitude of Y_c and delayed by six sample periods.

Strip-chart recordings, hard copy output from CONSIM, and a snapshot of the display of U_d ~~is~~ are included in the following pages. The save-on-input option was not used in this example.

Example 2

Example 2 is taken from Reference 3. A digital controller implemented as

$$D(z) = 2.5\,\frac{1.\emptyset-\emptyset.5\emptyset3z^{-1}+\emptyset.\emptyset4968z^{-2}}{1.\emptyset-\emptyset.632z^{-1}-\emptyset.368z^{-2}}$$

is used as a deadbeat controller for the continuous time system described by the transfer function in the s-domain

$$G(s) = \frac{2.\emptyset}{(s+1)(s+2)}\,.$$

The first output shown in the following pages for this example is for an initial condition of 2.∅ volts. Here save-on-input is not desired since one would like to observe the response due to the initial condition on Y_c.

The next output for this example is for a step input of -2.∅ volts applied using the save-on-input option. N_I was set at 5 so the input was applied at the sixth sample time. Note on the hard copy output of the control that there is considerable (or at least noticeable) jitter in the control even before the input is applied. (Again it should be pointed out that at the time of writing this paper the data channel is not working and none of the results shown include hard copy output of Y_c. Hence, strip-chart recordings are included.) Since the controller is designed for deadbeat control, at each sample point after the third Y_c should be 2.∅ volts. This does not appear to be quite the case in the strip-chart recordings, but this is probably due to the extremely slow response of the strip-chart recorder used.

The final output for this example is with the coefficients rounded-off to the nearest tenth. Surprising little effect is seen in the response.

REFERENCES

1. Conant, Brian K., Design of a New Solid State Differential Analyzer Making Use of Integrated Circuits, Ph. D. Dissertation, University of Arizona, 1968.

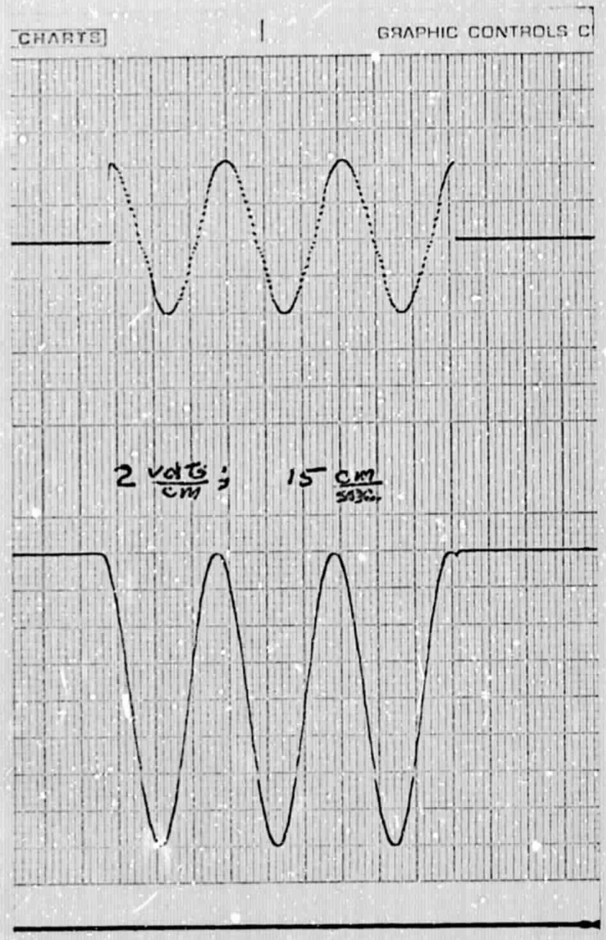
2. Wilkins, Jeff, A Modern Hybrid Computer Interface, Masters Thesis, University of Arizona, 1969.

3. Gupta, S. C. and Hansdorff, L., Fundamentals of Automatic Control, Example 9.5-1.

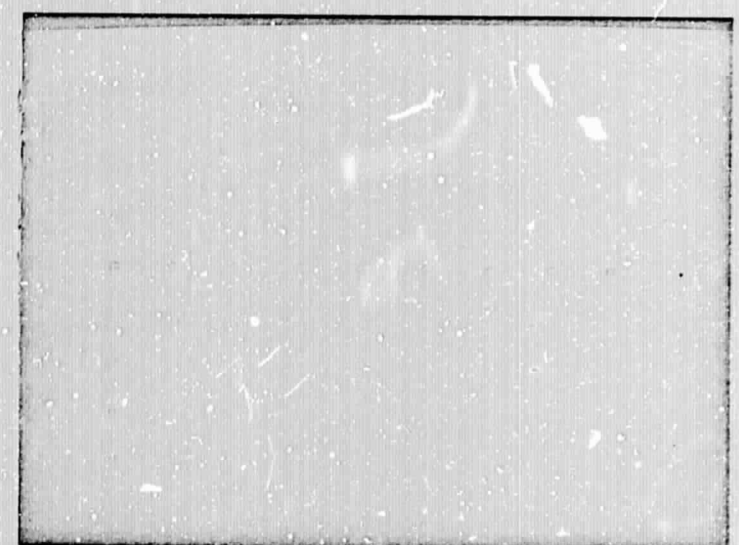
CONTROLLER COEFFICIENTS

A0	=	[illegible]	B0	=	0.999999
A1	=	[illegible]	B1	=	[illegible]
A2	=	[illegible]	B2	=	[illegible]
A3	=	[illegible]	B3	=	[illegible]
A4	=	[illegible]	B4	=	[illegible]
A5	=	[illegible]	B5	=	[illegible]
A6	=	0.5[illegible]	B6	=	[illegible]

Teletype Output of Controller Coefficients for Example 1.



Example 1. Simple delayed Sine-Loop. Top - Control, U_d Bottom - Output, Y_e.



Example 1. Delayed Sink-Loop. Picture of scope display of first 5ø points of U_d.

	CONTROL (IN VOLTS)	OUTPUT (IN VOLTS)
1	[illegible]	[illegible]
3	[illegible]	[illegible]
5	[illegible]	[illegible]
7	1.984	[illegible]
9	1.916	[illegible]
11	1.769	[illegible]
13	1.554	[illegible]
15	1.231	[illegible]
17	[illegible]	[illegible]
19	[illegible]	[illegible]
21	[illegible]	[illegible]
23	[illegible]	[illegible]
25	[illegible]	[illegible]
27	[illegible]	[illegible]
29	[illegible]	[illegible]
31	-1.574	[illegible]
33	-1.799	[illegible]
35	-1.945	[illegible]
37	-2.023	[illegible]
39	[illegible]	0.000
41	-1.926	0.000
43	[illegible]	0.000
45	-1.525	0.000
47	-1.241	0.000
49	[illegible]	0.000

Teletype Output of System

Response for Example 1

(Sine Loop)

CONTROLLER COEFFICIENTS

A0	=	[illegible].999999	B0	=	0.999999
A1	=	-[illegible].5[illegible]3[illegible]	B1	=	-0.632[illegible]
A2	=	[illegible].[illegible]49[illegible]	B2	=	-0.36[illegible]
A3	=	[illegible]	B3	=	0.[illegible]
A4	=	[illegible]	B4	=	0.[illegible]
A5	=	[illegible]	B5	=	[illegible]
A6	=	[illegible]	B6	=	[illegible]

Controller Coefficients for

Example 2.

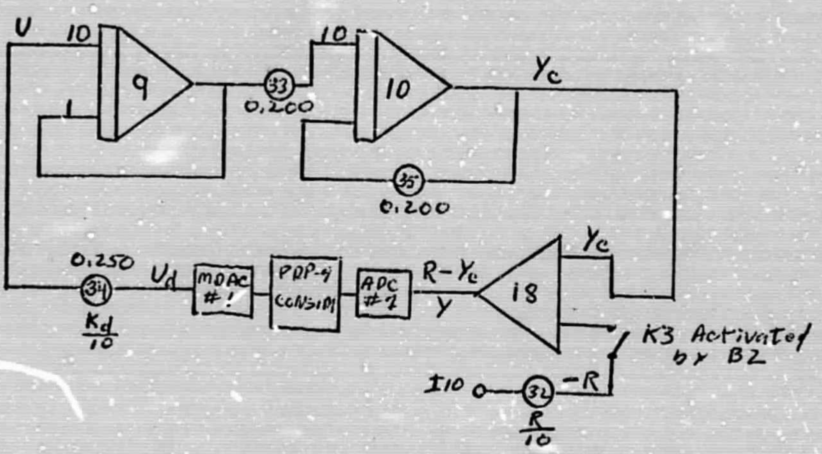
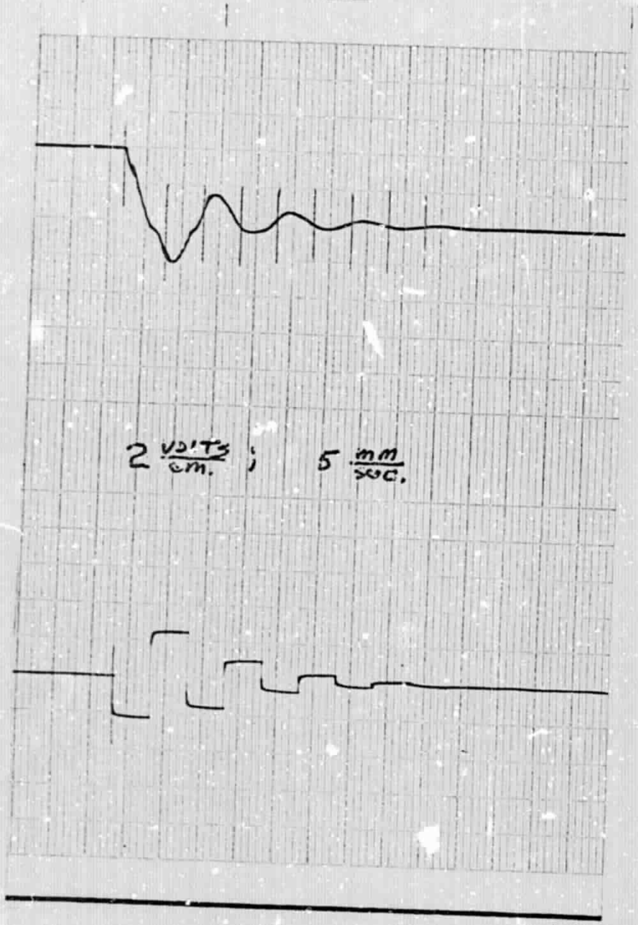
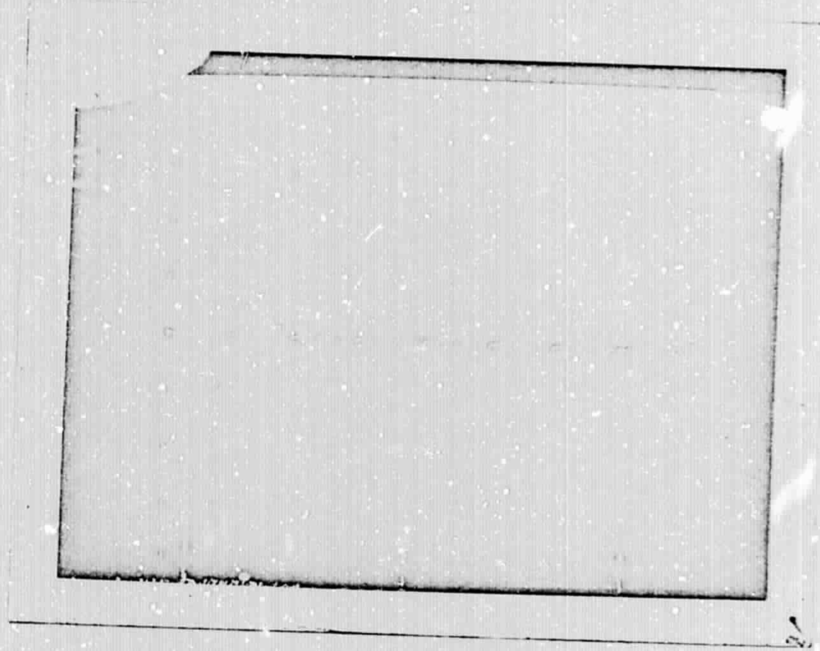


Fig. II Analog Simulated System for Example 2.



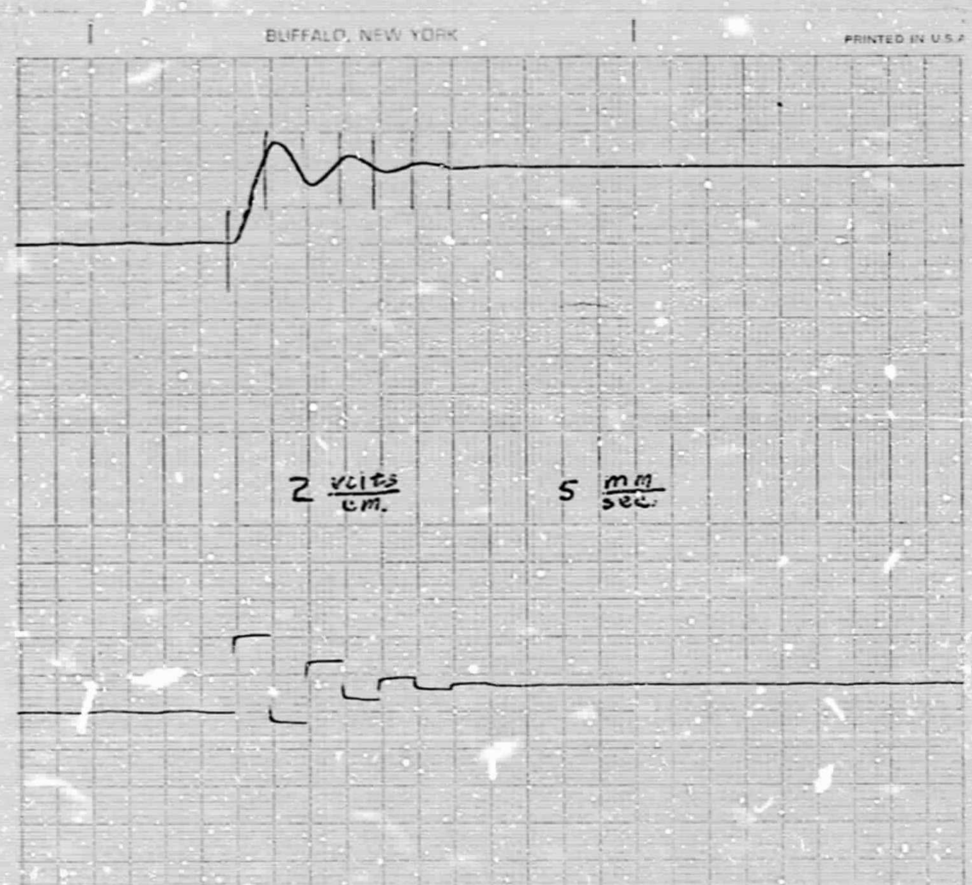
Example 2 under initial condition $Y_c(0) = 2.\phi$ volts. Save-on-input not used. Top - (-) Y_c Bottom - (-) U_d.



Example 2 under initial condition $Y_c(0) = 2.0$ volts. Display of first 25 points of U_d.

	CONTROL (IN VOLTS)	OUTPUT
1	1.3001	0.0000
2	-1.4370	0.0000
3	0.9791	0.0000
4	-0.5963	0.0000
5	0.3128	0.0000
6	-0.2240	0.0000
7	0.0973	0.0000
8	-0.0[illegible]	0.0000
9	0.0[illegible]	0.0000
10	-0.0293	0.0000
11	-0.0196	0.0000
12	-0.0098	0.0000
13	-0.0098	0.0000
14	0.0000	0.0000
15	-0.0195	0.0000
16	0.0000	0.0000
17	-0.0196	0.0000
18	0.0000	0.0000
19	-0.0196	0.0000
20	0.0000	0.0000
21	-0.0196	0.0000
22	0.0000	0.0000
23	-0.0196	0.0000
24	-0.0098	0.0000
25	0.0098	0.0000

System response for I.C. in Example 2. (exact coefficients)



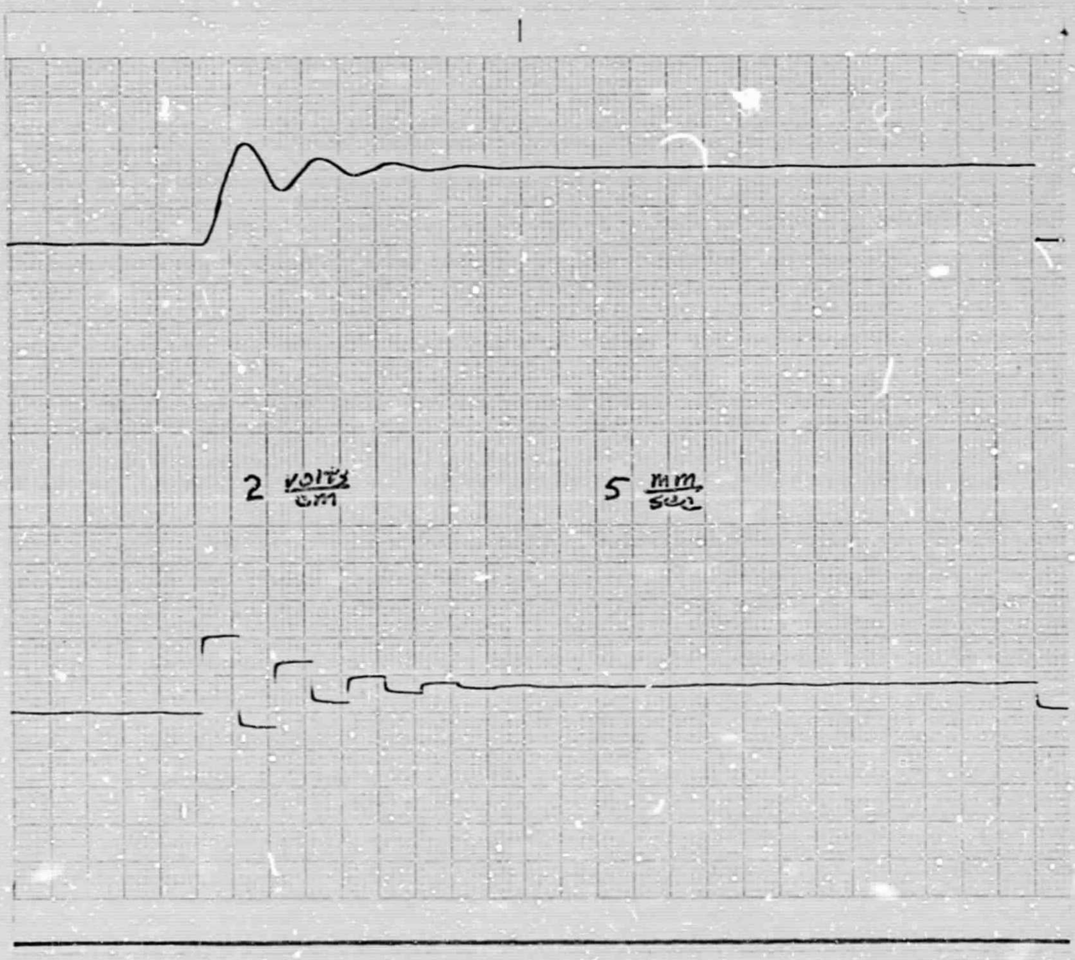
Example 2 with input of -2.0 volts (step).
Save-on-input used, $N_I = 5$. Top - $(-)Y_c$
Bottom - $(-)U_d$

	CONTROL	OUTPUT
	(IN VOLTS)	
1	0.0098	0.0000
2	-0.7293	0.0000
3	0.0098	0.0000
4	-0.2293	0.0000
5	[illegible]	0.0000
6	-0.3751	0.0000
7	0.3401	0.0000
8	-1.5640	0.0000
9	-0.3617	0.0000
10	-1.2459	0.0000
11	-0.5647	0.0000
12	-0.2602	0.0000
13	-0.7022	0.0000
14	-0.3016	0.0000
15	-0.7918	0.0000
16	-0.7913	0.0000
17	-0.8016	0.0000
18	-0.7821	0.0000
19	-0.8016	0.0000
20	-0.7820	0.0000
21	-0.8016	0.0000
22	-0.7820	0.0000
23	-0.8016	0.0000
24	-0.7821	0.0000
25	-0.8113	0.0000

System response for step
input in Example 2.
(exact coefficients)

CONTROLLER COEFFICIENTS

A0 = 0.999999	B0 = 0.999999
A1 = -0.5[illegible]	B1 = -0.6[illegible]
A2 = 0.[illegible]	B2 = -0.4[illegible]
A3 = 0.[illegible]	B3 = 0.[illegible]
A4 = 0.[illegible]	B4 = 0.[illegible]
A5 = 0.[illegible]	B5 = 0.[illegible]
A6 = 0.[illegible]	B6 = 0.[illegible]



Coefficients and strip-chart output for step input to Example 2 with coefficients rounded to nearest tenth.

	CONTROL (IN VOLTS)	OUTPUT
1	0.0000	0.[illegible]
2	-0.2196	0.[illegible]
3	0.0000	0.[illegible]
4	-0.2196	0.[illegible]
5	0.0000	0.[illegible]
6	-2.3949	0.[illegible]
7	0.4399	0.[illegible]
8	-1.5236	0.[illegible]
9	-0.3123	0.[illegible]
10	-1.1144	0.[illegible]
11	-0.3061	0.[illegible]
12	-0.9139	0.[illegible]
13	-0.7331	0.[illegible]
14	-0.8309	0.[illegible]
15	-0.7722	0.[illegible]
16	-0.8616	0.[illegible]
17	-0.7918	0.[illegible]
18	-0.7820	0.[illegible]
19	-0.8143	0.[illegible]
20	-0.7722	0.[illegible]

System response for step input in Example 2 with rounded coefficients.

APPENDIX A

LISTING OF CONSIM

```
           .TITLE CONSIM
/MACRO-9 PROGRAM TO DIRECTLY IMPLEMENT A DIGITAL CONTROLLER
/  IN THE PDP-9 TO BE USED IN A CLOSED LOOP WITH A CONTINUOUS
/  TIME SYSTEM SIMULATED IN LOCUST, A FAST HYBRID COMPUTER.
/  THE Z-TRANSFORM COEFFICIENTS OF THE CONTROLLER TRANSFER
/  FUNCTION ARE INPUT THROUGH THE FORTRAN SUBROUTINE DIGIN.
/  THE PROGRAM IS WRITTEN TO OPERATE AS FAST AS POSSIBLE AND
/  SHOULD HANDLE REAL-TIME SAMPLING RATES EASILY AS HIGH AS
/  1 KHZ. FOR LOW ORDER SYSTEMS. UP TO 5TH ORDER SYSTEMS MAY
/  BE ACCOMODATED AT LOWER SAMPLING RATES.
/
/          EE 342 PROJECT
/          SUMMER, 1970
/          W. F. MOORE
/
           .GLOBL DIGIN
           .IODEV 2,3
/
/MACRO FOR FORMING PRODUCTS OF THE FORM A*Y(N) AND
/  ACCUMULATING THE PRODUCT IN AIT.
           .DEFIN .MULT,CONST,VARI
           LAC VARI
           GSM
           DAC .+3
           LAC CONST
           MULS
           0
           LLS 1
           ADD AIT; DAC AIT
           .ENDM
/
/ ROUTINE FOR TERMINATION BY FREE FLAG 0.
STOP       733444    /CLEAR FF0.
           701647+10 /TURN OFF LOCUST. (CL4+701647)
           LAD1F     /CLEAR ALL ADC FLAGS IN CASE
           LAD2F     /  THEY ARE UP.
           LAD3F
           LAD4F
           LAC (L!LDAIX+10       /CLEAR MDAC#1 TO ZERO.
           LAW -1; DAC RUNFLG#            /SET FLG TO ENTER RDRUN.
           DZM RFLG /CLEAR FLAG IN RERUN ROUTINE.
SETC       .TRAN 3,3,L22,ONE,             /SET CURSOR TO LINE 22
           LAW -2    / ALLOW RETURN FROM DISPL TO
           DAC IDX#  / QUERY FOR NEW COEFFICIENTS AGAIN,
           LAC (JMS DISPL     /  THEN GO TO RERUN.
           DAC SETJMP
           .WRITE 3,2,MES1-2,0            /RUN TERMINATED.
           .WAIT 3
```

```
        .WRITE 3,2,MES11-2,0              /NEW R-T OPTIONS?
        .READ 2,2,INBUF,4
        .WAIT 2
        LAC INBUF+2        /DECODE ANSWER TO .READ.
        AND MASK1
        SAD ASCY  /YES?
        JMS OPT
        SAD ASCN  /NO?
        JMP SETC2
        JMP SETC  /WRONG ANSWER.
SETC2   .T  3,3,122,ONE,
        LAC (JMP RETUN      / CHANGE RETURN FROM DISPL
        ISZ IDX             / THE SECOND TIME AROUND.
LSKP    SKP
        DAC SETJMP
        .WRITE 3,2,MES12-2,               /$$
        .WAIT 3
        .WRITE 3,2,MES2-2,                /NEW COEFFICIENTS?
        .READ 2,2,INBUF,4
        .WAIT 2
        LAC INBUF+2         /DECODE ANSWER.
        AND MASK1 /MASK FIRST 7 BITS FOR FIRST ASCII CHARACTER.
        SAD ASCY  /ASCII Y.
        JMP INST  /RETURN TO DIGIN FOR NEW COEFFICIENTS.
        SAD ASCN  /ASCII N.
SETJMP  JMS DISPL /QUERY FOR DISPLAY AND HARD COPY.
                  / SECOND TIME ENTERED - JMP RETUN
        JMP SETC2 /WRONG ANSWER TO .READ OR FIRST RETURN FROM DISPL.
/
/
/BEGIN COEFFICIENT INPUT.
START   LAC TRUE
        DAC INIT# /MAKE INIT = .TRUE.
        .INIT 2,2,START    /DEVICE INITIALIZATION FOR CRT 1-0.
        .INIT 3,1,START
        JMS OPT   /PICK RUN OPTIONS.
INST    LAC L.16  /INDEX FOR 14(DEC) ARGUMENTS.
        DAC COUNT#
        LAC TRUE  /MAKE INPUT .TRUE.
        DAC INPUT#
        LAC LAC:  DAC  IP# /INITIALIZE COEFFICIENT POINTER.
        LAC (CSM  /SET UP FIRST LOCATIONS OF .MULT MACRO
        DAC ACY   / EXPANSIONS IN CASE THEY HAVE BEEN
        LAC (LAC YN.1       /CHANGED BY THE DELETE SUPR.
        DAC A1Y
        LAC (LAC YN.2
        DAC A2Y
        LAC (LAC YN.3
```

```
         DAC A3Y
         LAC (LAC YN.4
         DAC A4Y
         LAC (LAC YN.5
         DAC A5Y
         LAC (LAC YN.6
         DAC A6Y
         LAC (LAC UN.1
         DAC B1U
         LAC (LAC UN.2
         DAC B2U
         LAC (LAC UN.3
         DAC B3U
         LAC (LAC UN.4
         DAC B4U
         LAC (LAC UN.5
         DAC B5U
         LAC (LAC UN.6
         DAC B6U
         JMS SUBR  /INPUT COEFFICIENTS.
         DZM INIT  /INITIAL COEFFICIENTS READ IN.
/
/UN-NORMALIZE THE COEFFICIENTS TO INTEGERS.
SCAL     LAC* APRA /GET EXPONENT.
         AND MASK2; TAD MASK3; XOR MASK3         /FORM 18 BIT EXP.
         CMA;      TAD ONE  /NEGATE IT.
         DAC AIT#
         TAD LASI7          /CHECK FOR A SHIFT THAT'S
         SMA!SZA!CLL        /TOO LONG.
         JMP OVERS
         LAC AIT
         TAD LLRS  /FORM SHIFT COMMAND.
         SKP
OVERS    LAC LRSTB
         DAC SHIFT
         ISZ APRA
         LAC* APRA /GET MANTISSA.
         RCL;      SZL      /MAKE ONES COMPLEMENT
         CMA;      RAR      /IF NEGATIVE.
SHIFT    XX        /PERFORM SHIFTING TO UN-NORMALIZE.
         DAC* AIP; ISZ AIP  /SAVE IN PROPER COEF. LOCATION.
         ISZ APRA
         ISZ COUNT
         JMP SCAL
/
         LAC LAP
         TAD (2             /IN .MULT MACROS USING ZERO
         DAC AIP            / COEFS., REPLACE THE FIRST
```

```
        LAC L.IG             / INSTRUCTION WITH   JMP TO
        DAC COUNT            / THE NEXT .MULT MACRO.
        LAC (MPT             /  THIS ELIMINATES MULTIPLICATION
        DAC TPTR#            /  BY ZERO.
BKDEL   LAC* MIP
        SNA
        JMS DELETE           /ZERO COEF. FOUND.
        ISZ MIP
        ISZ TPTR
        ISZ COUNT
        JMP BKDEL
        LAC RUNFLG           /CHECK TO SEE IF RUNTIME OPTS
                             /  HAVE BEEN CHANGED.
        SNA
        JMP SIM              /THEY HAVE, BEGIN SIMULATION.
        LAW -1               /THEY HAVEN'T, SO RUN WILL HAVE
        DAC REFLG#           /TO BE STARTED BY RERUN ROUTINE.
        JMP RERUN
/
DELETE  0          /SUBR. TO JMP AROUND .MULT MACROS WITH 0 COEFS.
        LAC* TPTR
        DAC TPTR2
        LAC TPTR
        DAC TPTR3#
        ISZ TPTR3
        LAC* TPTR3
        AND (17777           /MASK ADDRESS TO MAKE RELATIVE.
        TAD (JMP
        DAC* TPTR2#
        JMP* DELETE
/
/ESTABLISH OPTIONS.
OPT     0
        .TRAN 3,3,ONE,ONE,1
        DZM RUNFLG           /CLEAR FLAG TO ENTER RERUN.
        .WRITE 3,2,MES4-2,2             /RUN-TIME OPTIONS.
        .WAIT 3
/
ZOVFL   .TRAN 3,1,L8,ONE,1
        .WRITE 3,2,MES5-2,2             /ALLOW DIGITAL OVERFLOW?
        .READ 2,2,INBUF,4
        .WAIT 2
        LAC INBUF+2
        AND MASKI
        SAD (SCY  /YES?
        JMP NOOVF
        SAD (SCN  /NO?
        JMP OVF
```

```
        JMP ZOVF   /WRONG ANSWER.
OVF     LAC (DAC AIP
        DAC OVFCK
        JMP ZSAVE
NOOVF   LAC (JMP LDAC
        DAC OVFCK
/
ZSAVE   .WRITE 3,2,MES6-2,0           /SAVE SOLUTION?
        .READ 2,2,INBUF,4
        .WAIT 2
        LAC INBUF+2
        AND MASK1
        SAD ASCY
        JMP YESSAV          /SAVE THE CTRL AND OUTPUT.
        SAD ASCN
        JMP NOSAV           /DON'T SAVE EITHER.
        JMP ZSAVE           /WRONG ANSWER.
NOSAV   LAC LNOP
        DAC SAVE
        LAC (JMP AIY
        DAC SVSKP
        LAC (JMP REG
        DAC SAVTES
        DZM SAVFLG
        LAC ASCN   /BYPASS RETURN TO OPT IN STOP.
        JMP* OPT
YESSAV  .WRITE 3,2,MES7-2,0           /HOW MANY POINTS?
        .READ 2,2,INBUF,4
        .WAIT 2
        JMS UNPACK          /UNPACK ANSWER TO OCTAL NUMBER.
        JMP YESSAV          /ILLEGAL ANSWER RETURN PT.
        JMP YSAV2           /LEGAL RETURN POINT.
/
/SUBROUTINE TO UNPACK AN ASCII DECIMAL # IN INBUF
/  TO AN OCTAL # IN IPOINT. CHECKS FOR A POSITIVE #
/  LESS THAN OR EQUAL TO 252(DEC.).
/       JMS UNPACK
/       (ILLEGAL RETURN)
/       (LEGAL RETURN POINT)
UNPACK  0
        DZM IPOINT#
        LAC INBUF+3
        LMQ                 /2ND ASCII WORD IN MQ.
        LAC INBUF+2         /1ST  "    "  "  AC.
SVLOOP  AND MASK1           /MASK FIRST DIGIT.
        CLL
        RTR; RTR; RTR       /RIGHT JUSTIFY DIGIT
        RTR; RTR; RTR       /IN AC.
```

```
        DAC AIT
        SAD (15     /CHECK FOR END OF LINE. (CARRIAGE RET.)
        JMP OUTSV
/ CHECK TO SEE THAT THE CHARACTER IS A NUMBER.
        TAD (-60
        SPA;        JMP NONUM
        TAD (-72+60
        SMA;        JMP NONUM
/ CONVERT THE ASCII DIGIT TO OCTAL.
        LAC AIT
        AND (17
        DAC AIT
        LAC IPOINT
        CLL
        RAL; RAL    /MULTIPLY PREVIOUS DIGIT
        TAD IPOINT          /   BY 10.
        TAD IPOINT
        TAD AIT     / ADD THIS DIGIT.
        DAC IPOINT
UT CURRENT LAC INBUF+2      /SHIFT
        LLS 7               /CHARACTER AND LEFT
        DAC INBUF+2         /JUSTIFY NEXT ONE.
        JMP SVLOOP
NONUM   .WRITE 3,2,MES8-2,2            /NOT A NUMBER.
        .WAIT 3
        JMP* UNPACK
NUMTB   .WRITE 3,2,MES9-2,2            /NUMBER TOO BIG.
        .WAIT 3
        JMP* UNPACK
OUTSV   LAC IPOINT
        TAD (-373 /CHECK FOR NUMBER GREATER THAN 251(10).
        SMA;        JMP NUMTB
        ISZ UNPACK          /LEGAL RETURN.
        JMP* UNPACK
/
YSNR    LAC IPOINT
        CMA;        TAD ONE  /T.C. POINT FOR COUNTER.
        DAC POINT
        DAC POINTS#
        LAC (DAC* 12        /SET UP TO SAVE THE CONTROL.
        DAC SAVE
        LAC (CTRL-1         /CONTROL ARRAY POINTER.
        DAC* (12            /USE AUTO INDEX REG. 12.
        LAC POINT# /SET UP WORD COUNT FOR DATA CHANNEL
        DAC* (32   /USE 32 AS W.C. REG.
        LAC (OUTPUT-1       /OUTPUT ARRAY POINTER.
        DAC* (33
        LAW -1     /SET SAVE FLAG.
```

```
            DZM YN.2; DZM YN.3
            DZM YN.4; DZM YN.5
            DZM YN.6; DZM UN.0
            DZM UN.1; DZM UN.2
            DZM UN.3; DZM UN.4
            DZM UN.5; DZM UN.6
/
/***** BEGINNING OF ACTUAL SIMULATION LOOP. *****
/
/INPUT FROM LOCUST.
ADC         LASIS     / ADC SKIP INSTRUCTION.
            JMP FIRST /IF ADC NOT READY TEST FOR EFR TO TERMINATE RUN.
            LADI      /READ ADC.
            LRSS 7    /RIGHT JUSTIFY ADC INPUT.
            SPA
            TAD L.1   /ONES COMPLEMENT IF NEGATIVE.
            DAC YN.0
/ FORM A2*Y(N-2)
AY          CSM
            DAC .+3
            LAC A2
            MULS
            0
            LLS 1     /ELIMINATE EXTRA SIGN FROM MULS.
/
/FORM CONTROL (UN.0) AND OUTPUT IT.
FCONT        ADD AIT   /ADD ACCUMULATED PRODUCTS.
             DAC AIT   /SAVE IN ACCUMULATOR REG.
             SPA; TAD ONE       /TWOS COMP. FOR ADAC OUTPUT.
OVFCK        DAC AIP
             TAD (-2...         /CHECK FOR OVERFLOW IF
             SMA                /CALLED FOR BY USER, OTHERWISE
             JMP OVERR          /OVFCK = JMP LDAC.
             TAD (2...+2...
             SPA
             JMP OVERR
             LAC AIP
LDAC         LLSS 7    /LEFT JUSTIFY FOR ADAC OUTPUT.
             LDALILDAX          /LOAD AND XFER. ADAC#1 BUFFER REG.
             LAC AIP
SAVE         DAC I2    /CHANGED TO NOP IF NO SAVE.
             CMA       /COMPLEMENT CURRENT CONTROL
             TAD ONE   /FOR FORMING NEXT CONTROL.
             DAC UN.
/
/CHECK FOR END OF CONTROL SAVING.
SVSKP        ISZ POINTS
             JMP SAMTES
```

```
         DZM YN.2; DZM YN.3
         DZM YN.4; DZM YN+5
         DZM YN.6; DZM UN.0
         DZM UN.1; DZM UN.2
         DZM UN.3; DZM UN.4
         DZM UN.5; DZM UN.6
/
/***** BEGINNING OF ACTUAL SIMULATION LOOP. *****
/
/INPUT FROM LOCUST.
ADC      LASIS    /ADC SKIP INSTRUCTION.
         JMP FIRST /IF ADC NOT READY TEST FOR FR2 TO TERMINATE RUN.
         LADI     /READ ADC.
         LRSS 7   /RIGHT JUSTIFY ADC INPUT.
         SPA
         TAD L.1  /ONES COMPLEMENT IF NEGATIVE.
         DAC YN.0
/ FORM A2*Y(N-3)
ADY      CSM
         DAC .+3
         LAC A2
         MULS
         
         LLS 1    /ELIMINATE EXTRA SIGN FROM MULS.
/
/FORM CONTROL (UN.0) AND OUTPUT IT.
FCONT    ADD AIT  /ADD ACCUMULATED PRODUCTS.
         DAC AIT  /SAVE IN ACCUMULATOR REG.
         SPA; TAD ONE        /TWOS COMP. FOR ADAC OUTPUT.
OVFCK    DAC AIP
         TAD (-2...          /CHECK FOR OVERFLOW IF
         SNA                 /CALLED FOR BY USER, OTHERWISE
         JMP OVERP           /OVFCK = JMP LDAC.
         TAD (2...+2...
         SPA
         JMP OVERM
         LAC AIP
LDAC     LLSS 7   /LEFT JUSTIFY FOR ADAC OUTPUT.
         LDA LILDAX          /LOAD AND XFER. ADAC#1 BUFFER REG.
         LAC AIP
SAVE     DAC* 12  /CHANGED TO NOP IF NO SAVE.
         CMA      /COMPLEMENT CURRENT CONTROL
         TAD ONE  /FOR FORMING NEXT CONTROL.
         DAC UN.
/
/CHECK FOR END OF CONTROL SAVING.
SVSKP    ISZ POINTS
         JMP SAVTES
```

```
          L C L  P
          D C S  VE
          L C (JMP .+2
          D C SVSKP
/
/IF S  E   INPUT, THIS TESTS FF  TO ST  T S VE  ND  PPLY IN UT.
S VTES    7  52      /FF1 TEST SKIP.
          JMP  EG
          ISZ  IM
JD        JMP DOWN
          L C (74          /APPLY THE C NTR L THROUGH
          7  647           / CON ROL REG. BIT 1. (BITS 2 & 3
                           /  LSO  ROVIDED HE E IF NEEDED.)
)         L C (JMP  EG     /BYP SS  FTER THIS.
          D C S VTES
          JMP .+3
DOWN      LAC LSKP  /SET UP FOR SAVE
          D C S VTES
          7 3544    /CLEA  FF1.
          LAC (D C* 12
          D C SAVE
          LAC (ISZ POINTS
          DAC SVSKP
          LAC P INTS
          DAC* (32
          LAC (OUTPUT-1
          DAC* (33
          LAC (JMP  EG
          DAC JD
          DAC DOWN-1
/
/REGISTER ADJUSTMENT FOR DELAY SIMUL TION.
 EG       L C YN.5#; D C XN.6#
          L C YN.4#;  AC YN.5
          L C YN.3#; D C YN.4
          L C YN.2#; DAC YN.3
          L C YN.1#;  AC YN.2
          L C YN.0#; D C YN.1
          LAC UN.5#;  AC UN.6#
          L C UN.4#;   C UN.5
          L C UN.3#; D C UN.4
          L C UN.2#;   C TN.3
          L C UN.1#; D C UN.2
          L C UN.0#; D C UN.1
          DZM  LT    /ZERO THE PRODUCT  CCUMUL TOR REG.
/
/FORM OTHER NEEDED PRODUCTS FOR NEXT CONTROL.
/
```

```
/ A1*Y(N-1)
A1Y        .MULT A1,YN.1
/
/ B1*U(N-1)
B1U        .MULT B1,UN.1
/
/ A2*Y(N-2)
A2Y        .MULT A2,YN.2
/
/ B2*U(N-2)
B2U        .MULT B2,UN.2
/
/ A3*Y(N-3)
A3Y        .MULT A3,YN.3
/
/ B3*U(N-3)
B3U        .MULT B3,UN.3
/
/ A4*Y(N-4)
A4Y        .MULT A4,YN.4
/
/ B4*U(N-4)
B4U        .MULT B4,UN.4
/
/ A5*Y(N-5)
A5Y        .MULT A5,YN.5
/
/ B5*U(N-5)
B5U        .MULT B5,UN.5
/
/ A6*Y(N-6)
A6Y        .MULT A6,YN.6
/
/ B6*U(N-6)
B6U        .MULT B6,UN.6
/
           JMP ADC
/
/***** END OF ACTUAL SIMULATION LOOP. *****
/
/OVERFLOW ROUTINES
OVERP      LAC (1777 /POSITIVE OVERFLOW CAUSES SATURATION AT 1777.
           DAC AIT
           JMP LDAC
OVERM      LAC (-1777 /NEG. OVERFLOW CAUSES SAT. AT -1777.
           DAC AIT
           JMP LDAC
/
```

```
/SUBROUTINE TO CALL DIGIN. IF INPUT=.TRUE., INPUT COEFS; IF
/  INPUT=.FALSE., CHECK FOR DISPLAY AND TTY OUTPUT.
SUB       XX
          LAC LCOEF; DAC ARRA /INITIALIZE INPUT ARRAY POINTER.
          JMS* DIGIN
          JMP .+7   /JMP ARGUMENTS.
          IN1
          INPUT
          SAVFLG
          
          
          3
          JMP* SUB
/
/SUBROUTINE TO TEST RFØ TO TERMINATE RUN.
RTEST     701541
          JMP ADC   /IF NO TERM., GO BACK AND TEST ADC.
          JMP STOP  /RFØ=1, TERMINATE.
/
/QUERY FOR DISPLAY AND TTY OUTPUT.
DISPL     0
          DZM INPUT
          JMS SUB
          LAC SAVFLG /IF SAVFLG UP, USER MAY WISH NEW
          SNA        /COEFS AFTER SEEING DISPLAY, IF NOT HE
          JMP RERUN /MAY WISH TO RERUN.
          JMP* DISPL
/
/STAND-BY TO RERUN UNDER SAME COEFFICIENTS.
RERUN     .TRAN 3,3,L22,ONE,1              /CURSOR TO LINE 22.
          LAC SAVFLG            /IF SAVE, RE-SETUP TO DO SO.
          SNA
          JMP REWRT
          LAC (CTRL-1
          DAC* (13
          DAC* (12
          LAC POINT
          DAC POINTS
          DAC* (32
          LAC (OUTPUT-1
          DAC* (14
          DAC* (33
          LAC (ISZ POINTS
          DAC SVSKP
          LAC (DAC* 12
          DAC SAVE
          LAC (-373 /ZERO THE SAVE ARRAYS.
          DAC IDX
```

```
ZE      DZM* 13
        DZM* 14
        ISZ IDX
        JMP ZE
        LAC INPFLG          /IF SAVE-ON-INPUT, RE-SETUP.
        S
        JMP EW T
        L C ITIME
          C TIME
        L C (7 152
        D C S VTES
        L C L  P
        D C SAVE
        L C (JMP DOWN
        D C JD
        LAC (JMP DOWN+2
        D C DOWN-1
        L C (JMP S VTES
        D C SVSKP
EW T    L C EFLG /IF ENTERED  FTER INPUTING COEFS. FROM DIGIN,
                  / EFLG=-1 AND THE FOLLOWING MESSAGE IS
                  / NOT NEEDED.
        SZ
        JMP SIM
        .WRITE 3,2,MES3-2,
        . EAD 2,2,INBUF,4
        .WAIT 2
        LAC INBUF+2;        AND MASKI
        S D SCR; JMP SIM  / UN?
        LAC INBUF+2
        S D ASCNO; JMP EXIT. /NO?!THEN EXIT TO MONITOR.
        JMP
EXIT.   .T   3,3,L ,L ,1  /CLE   THE SCREEN
        .EXIT     /RETU N TO MONITOR.
/
/   Y DESC IPTO  BLOCKS.
/   C EFFICIE    Y.
          +34 /   L     Y, 2 (1 ) W  DS LONG.


        COEFS
/   CONT OL    Y.
        +37       /INTEGE       Y, 25 (1 ) WO DS LONG.


        CTRL
/   UTPUT    Y
         +37
```

```
        0
        0
 [illegible]3    OUTPUT
/
/LITERALS
L[illegible]    0
ON[illegible]   1
L3      1[illegible]
L11     1[illegible]
L22     22
L.1     -1
L.16    -16
M SK1   774[illegible]
M SK2   777
MASK3   7774[illegible]
ASCY    544[illegible]    /LEFT JUSTIFIED ASCII Y.
ASCN    47[illegible]     / "      "       "    N.
ASC     51[illegible]     / "      "       "    R.
ASCN[illegible]  .ASCII /NO/<015>   /NO FOR RETURN TO MONITOR.
TRUE    777777
LON     4[illegible]
L SI7   777761
LL[illegible]S)  L SS
L STB   LESS+21
LNOP    NOP
LA0     A0
LCOEF   COEFS
/
/COEFFICIENT [illegible]Y (SCALED).
A0      0
B0      0
A1      0
B1      0
A2      0
B2      0
[illegible]3      0
B3      0
[illegible]4      0
B4      0
A5      0
B5      0
[illegible]6      0
B6      0
/
/POINTE[illegible]Y TO .MULT M CROS. USED IN THE PROCESS OF
/ DELETING UNUSED MULTIPLICATIONS (THOSE WITH ZERO COEF).
MPT     A1Y
        B1U
```

```
        A2Y
        B2U
        A3Y
        B3U
        A4Y
        B4U
        A5Y
        B5U
        A6Y
        B6U
        DC
/
/OUTPUT MESSAGES.
MES1    .ASCII /$$ RUN TERMINATED:/<175>
MES2    .ASCII / NEW COEFFICIENTS? $$/<15>
MES3    .ASCII /$$ TYPE RU TO RESTART, NO TO EXIT $$/<15>
MES4    .ASCII /        $$ SIMULATION UNDERWAY $$/<15>
MES5    .ASCII /ALLOW DIGITAL OVERFLOW? /<175>
MES6    .ASCII / SAVE THE SOLUTION VALUES? /<175>
MES7    .ASCII / HOW MANY POINTS (.LT. 250)? /<175>
MES8    .ASCII / NOT A NUMBER!/<15>
MES9    .ASCII / TOO MANY POINTS!/<15>
MES10   .ASCII /         $$ RUN-TIME OPTIONS $$/<15>
MES11   .ASCII / NEW R-T OPTIONS? $$/<15>
MES12   .ASCII /$$/<175>
MES13   .ASCII / SAVE ON INPUT? /<175>
MES14   .ASCII / INPUT AFTER N SAVED SAMPLES: N = /<175>
/
/IO BUFFERS
COEFS   .BLOCK 34 /STORAGE FOR REAL COEFS.
CTRL    .BLOCK 373 /THE CONTROL IS SAVED IN THIS BUFFER.
OUTPUT  .BLOCK 373 /THE OUTPUT IS SAVED HERE.
INBUF   .BLOCK 52 /INPUT LINE BUFFER FOR GET.
        .END START
```

APPENDIX B

LISTING OF DIGIN

```
C   SUBROUTINE TO HANDLE COEFFICIENT
C   INPUT AND DISPLAY AND TTY
C   OUTPUT FOR CONSIM.
C
C        EE 342 PROJECT
C        SUMMER, 1970
C        W. F. MOORE
C
        SUBROUTINE DIGIN (INIT,INPUT,
      #SVFLG,COEFS,ICONT,IOUTP)
        LOGICAL INIT,INPUT,SVFLG,QUEB
        DIMENSION COEFS(14),ICONT(250)
        DIMENSION IOUTP(250),ISAVE(250)
        DIMENSION FOUTP(250),FCONT(250)
        DATA EH0/2HA0/,BE0/2HB0/,EH1/2HA1
      #/,BE1/2HB1/,EH2/2HA2/,BE2/2HB2/,
      #EH3/2HA3/,BE3/2HB3/,EXIT/2HRU/,STP
      #/1HN/,YEP/1HY/,CONT/1HC/,OUTP/1HO/
      #,EH4/2HA4/,BE4/2HB4/,EH5/2HA5/,BE5
      #/2HB5/,EH6/2HA6/,BE6/2HB6/
        QUEB<.FALSE.
        IF(.NOT.INPUT) GO TO 100
C
C   COEFFICIENT INPUT SECTION.
C   NOTE THE ORDER OF THE COEFFICIENTS:
C    COEF(1)=A0; COEF(2)=B0; COEF(3)=A1
C    COEF(4)=B1; ETC.
C
        IF(.NOT.INIT) GO TO 2
        DO 1 I=1,8
1       COEFS(I)=0.0
        COEFS(2)=0.999999
2       WRITE(3,3) (COEFS(I), I=1,14)
3       FORMAT(*-A0 = *,F9.6,5X,*B0 = *,F
      #9.6//* A1 = *,F9.6,5X,*B1 = *,F9.6
      #//* A2 = *,F9.6,5X,*B2 = *,F9.6//*
      # A3 = *,F9.6,5X,*B3 = *,F9.6//* A4
      # = *,F9.6,5X,*B4 = *,F9.6//* A5 =
      # *,F9.6,5X,*B5 = *,F9.6//* A6 = *,
      #F9.6,5X,*B6 = *,F9.6)
        CALL SETCUR (3,1,18,0,1)
        IF(QUEB) GO TO 52
        IF (.NOT.INIT) GO TO 5
        WRITE(3,4)
4       FORMAT(* $$ INPUT COEFFICIENTS $$
      #*)
        GO TO 7
52      WRITE(3,53) COEFS(2)
```

```
53     FORMAT(* $$ B0 MUST BE *,F9.6,
      #*, TRY AGAIN $$*)
       QUEB=.FALSE.
       GO TO 7
5      WRITE(3,6)
6      FORMAT(* $$ CHANGE COEFFICIENTS $
      #$*)
7      READ(2,8) SPEC,DATA
8      FORMAT(A2,1X,F9.6)
       IF(DATA.GT.0.999999) GO TO 9
       IF(SPEC.EQ.EH0) COEFS(1)=DATA
       IF(SPEC.EQ.BE0) QUEB<.TRUE.
       IF(SPEC.EQ.EH1) COEFS(3)=DATA
       IF(SPEC.EQ.BE1) COEFS(4)=DATA
       IF(SPEC.EQ.EH2) COEFS(5)=DATA
       IF(SPEC.EQ.BE2) COEFS(6)=DATA
       IF(SPEC.EQ.EH3) COEFS(7)=DATA
       IF(SPEC.EQ.BE3) COEFS(8)=DATA
       IF(SPEC.EQ.EH4) COEFS(9)=DATA
       IF(SPEC.EQ.BE4) COEFS(10)=DATA
       IF(SPEC.EQ.EH5) COEFS(11)=DATA
       IF(SPEC.EQ.BE5) COEFS(12)=DATA
       IF(SPEC.EQ.EH6) COEFS(13)=DATA
       IF(SPEC.EQ.BE6) COEFS(14)=DATA
       ID(SPEC.EQ.EXIT) RETURN
       GO TO 2
9      CALL SETCUR (3,1,18,0,1)
       WRITE(3,14)
14     FORMAT(* $$ COEFFICIENT TO LARGE,
      # TRY AGAIN $$*)
       GO TO 7
C
C  DISPLAY AND TTY OUTPUT SECTION.
C
100    IF(.NOT.SVFLG) GO TO 280
       CALL SETCUR (3,3,18,0,1)
98     WRITE(3,101)
101    FORMAT(* $$ SELECT DISPLAY OR NON
      #E $$*)
97     READ(2,102) ANS
102    FORMAT(A1)
       IF(ANS.EQ.STP) GO TO 200
       CALL SETCUR (3,3,18,0,1)
       WRITE(3,103)
103    FORMAT(* $$ HOW MANY POINTS? $$*)
99     READ(2,104) NPOINT
104    FORMAT(I3)
       IF(NPOINT.GT.250) GO TO 105
```

```
      IF(ANS.EQ.CONT) GO TO 300
      IF(ANS.EQ.OUTP) GO TO 400
      CALL SETCUR (3,3,18,0,1)
      WRITE(3,96)
96    FORMAT(* $$ ILLEGAL VAR. NAME! TR
     #Y AGAIN $$*)
      GO TO 97
105   CALL SETCUR (3,3,18,0,1)
      WRITE(3,106)
106   FORMAT(* $$ TOO MANY POINTS! TRY
     # AGAIN $$*)
      GO TO 99
300   DO 301 I=1,NPOINT
301   ISAVE(I)=ICONT(I)
302   CALL ISCALE (ISAVE,NPOINT,ISHF)
      CALL DISPLA (0,0,ISAVE,NPOINT)
      CALL SETCUR(3,3,9,1,1)
      FULL=(10.0)/(2.0**(ISHF-7))
      WRITE(3,303) FULL
303   FORMAT(9X,* FULL SCALE = *,F4.2,
     #* VOLTS*/)
      CALL SETCUR(3,1,18,0,1)
      GO TO 98
400   DO 401 I=1,NPOINT
401   ISAVE(I)=IOUTP(I)
      GO TO 302
200   DO 199 I=1.250
199   ISAVE(I)=0.0
      CALL SETCUR (3,3,18,0,1)
      WRITE(3,201)
201   FORMAT(* $$ TELETYPE PRINT-OUT? $
     #$*)
      READ(2,102)ANS
      IF(ANS.EQ.STP) RETURN
      IF(ANS.NE.YEP) GO TO 200
250   CALL SETCUR(3,3,18,0,1)
198   WRITE(3,270)
270   FORMAT(* $$ DATA PRINT-OUT? $$*)
      READ(2,102)ANS
      IF(ANS.EQ.STP) GO TO 280
      IF(ANS.NE.YEP) GO TO 198
      CALL SETCUR (3,3,18,0,1)
      WRITE(3,103)
249   READ(2,104)NPOINT
      IF(NPOINT.GT.250) GO TO 262
998   CALL SETCUR (3,3,18,0,1)
      WRITE(3,999)
999   FORMAT(* $$ PRINT EVERY NTH POINT
```

```
     # $$*/*+ N = *)
       READ(2,104)INCR
       CALL SETCUR(3,3,9,1,1)
       WRITE(3,251)
251    FORMAT(9X,* $$ OUTPUT ON TTY $$*)
       WRITE(5,254)
254    FORMAT(1H1/////9X,* CONTROL*,9X,
     #*OUTPUT*/15X,* (IN VOLTS)*//)
       ICTR=0
       DO 252 I=1,NPOINT,INCR
       FCONT(I)=ICONT(I)
       FCONT(I)=FCONT(I)*10.0/1023.0
       FOUTP(I)=IOUTP(I)
       FOUTP(I)=FOUTP(I)*10.0/1023.0
       ICTR=ICTR+1
       IF(ICTR.NE.51) GO TO 252
       WRITE(5,254)
       ICTR=0
252    WRITE(5,253)I,FCONT(I),FOUTP(I)
253    FORMAT(I4,3X,F10.3,5X,F10.3)
       GO TO 280
260    CALL SETCUR(3,3,18,0,1)
       WRITE(3,106)
       GO TO 249
280    CALL SETCUR(3,3,18,0,1)
       WRITE(3,281)
281    FORMAT(* $$ COEFFICIENT PRINT-OUT
     #? $$*)
       READ(2,102)ANS
       IF(ANS.EQ.STP) RETURN
       IF(ANS.NE.YEP) GO TO 280
       CALL SETCUR (3,3,9,0,1)
       WRITE(3,251)
       WRITE(5,282)
282    FORMAT(1H1///5X,* CONTROLLER COEF
     #FICIENTS*///)
       WRITE(5,3) (COEFFS(I),I=1,14)
       RETURN
       END
```